# Task

Write an application that parses a binary input file and produces a tab separated ascii file.

# Input file

The input file is a fixed length binary file with no line breaks.

Fields encoded as integers are stored in Big-endian. The file contains one or more extra bytes prior to some of the integer fields. Those extra bytes all contain the hex value 0xE9. The extra bytes should be spooled past before the integer field is interpreted.

## Fields present in file

(A = Ascii, I = Integer)

| # | Field name | Length and Type |
| --- | --- | --- |
|  | subscriber\_no | A25 |
|  | Imsi | A20 |
|  | channel\_seizure\_dt | A14 |
|  | provider\_id | A6 |
|  | message\_switch\_id | A3 |
|  | call\_direction\_code | A1 |
|  | secondary\_no | A25 |
|  | units\_of\_measure | A1 |
|  | at\_call\_dur\_sec | I4 |
|  | call\_category | A1 |
|  | call\_source | A5 |
|  | call\_dest | A5 |
|  | basic\_service\_code | A10 |
|  | ss\_code\_1 | A6 |
|  | ss\_code\_2 | A6 |
|  | maf\_feature | A30 |
|  | friend\_ind | A1 |
|  | calling\_route\_ind | A1 |
|  | at\_rated\_ftr\_lvl | A1 |
|  | user\_group\_ind | A1 |
|  | special\_num\_ind | I4 |
|  | guide\_by | A1 |
|  | equipment\_no | A20 |
|  | product\_type | A1 |
|  | rate\_ind | A1 |
|  | air\_free\_code | A1 |
|  | location\_area | A11 |
|  | gd\_operator\_id | A6 |
|  | gd\_switch\_id | A20 |
|  | gd\_cell\_id | A5 |
|  | calling\_country\_code | A3 |
|  | calling\_country\_exc | A5 |
|  | called\_country\_code | A3 |
|  | called\_country\_exc | A5 |
|  | bl\_usg\_consider\_ind | A1 |
|  | bill\_presentation\_no | A25 |
|  | pac\_code | A5 |
|  | basic\_service\_type | A1 |
|  | secret\_number\_ind | A1 |
|  | orig\_net\_chg\_amt | I4 |
|  | eu\_orig\_net\_chg\_amt | I8 |
|  | rm\_tax\_amt\_air | I4 |
|  | rm\_amt\_air | I4 |
|  | pabx\_orig\_subscriber | A25 |
|  | in\_code | A1 |
|  | cdr\_source | A1 |
|  | call\_term\_code | A3 |
|  | roam\_ind | A1 |
|  | channel\_type | A5 |
|  | transp\_ind | A1 |
|  | ms\_classmark | A7 |
|  | ss\_param | A21 |
|  | original\_call\_type | A1 |
|  | original\_call\_npi | A5 |
|  | original\_call\_number | A25 |
|  | dual\_service\_type | A1 |
|  | dual\_service\_code | A2 |
|  | o\_sdr\_exchange\_rate | I4 |
|  | o\_sdr\_amount | I4 |
|  | chain\_reference | I4 |
|  | upmark\_pct | I4 |
|  | sdr\_limit | I4 |
|  | o\_outcol\_cycle\_code | I2 |
|  | at\_soc | A9 |
|  | at\_soc\_date | A8 |
|  | recycle\_ind | A1 |
|  | fluctuation\_ind | A1 |
|  | call\_way | A1 |
|  | Apn | A101 |
|  | charging\_id | A10 |
|  | uplink\_volume | I4 |
|  | downlink\_volume | I4 |
|  | gprs\_duration | I4 |
|  | intl\_mobile\_subid | A18 |
|  | vlr\_country\_code | A3 |
|  | vlr\_country\_exc | A5 |
|  | record\_type | A2 |
|  | orig\_file\_id | A12 |
|  | call\_treatment\_storage | A1 |
|  | call\_info | A1 |
|  | sg\_ind | A1 |
|  | call\_source\_detailed | A5 |
|  | feature\_selection\_dt | A14 |
|  | o\_pdp\_type | A1 |
|  | o\_pdp\_address | A15 |
|  | partial\_ind | A1 |
|  | gsm\_qosr\_delay | I4 |
|  | gsm\_qosr\_mean\_thru | I4 |
|  | gsm\_qosr\_peak\_thru | I4 |
|  | gsm\_qosr\_precedence | I4 |
|  | gsm\_qosr\_reliability | I4 |
|  | gsm\_qosu\_delay | I4 |
|  | gsm\_qosu\_mean\_thru | I4 |
|  | gsm\_qosu\_peak\_thru | I4 |
|  | gsm\_qosu\_precedence | I4 |
|  | gsm\_qosu\_reliability | I4 |
|  | o\_gd\_switch\_id2 | A20 |
|  | clir\_status\_ind | A1 |
|  | mp\_rec\_id | A9 |
|  | rating\_time | A14 |
|  | rating\_rec\_type | A2 |
|  | ocp\_traffic\_type | A2 |
|  | service\_code | A10 |
|  | content\_description | A40 |
|  | terminal\_type | A15 |
|  | o\_NABS\_number\_of\_units | I4 |
|  | o\_NABS\_user\_owner | A1 |
|  | o\_NABS\_lcp\_owner | A1 |
|  | revenue\_share\_id | A12 |
|  | subscription\_id | A10 |
|  | rating\_group | A10 |
|  | content\_specifier | A100 |
|  | utc\_offset | A5 |
|  | location\_area\_last | A11 |
|  | gd\_cell\_id\_last | A6 |

# Output file

The output file should be a tab-separated ascii file on this format:

<Value for Field 1>\t<Value for Field 2>\t…<Value for Field n>\n

<Value for Field 1>\t<Value for Field 2>\t…<Value for Field n>\n

.

.

.

<Value for Field 1>\t<Value for Field 2>\t…<Value for Field n>\n

All trailing spaces found in ascii fields should be trimmed away.

All date fields should be written on the form “YYYYMMDD HH24MISS”

# Requirements

Create an application that reads in the input file according to the specifications above and outputs all fields into a text file as per above.

The application should be written in C++

Neither STL, any class in the namespace std::, nor any third party library except the included xml library can be used. For file operations use C-style FILE\* F = fopen etc

Field definitions should be stored in and read from an xml file using the supplied third party xml library.

The xml file can be created manually.

This is a small function that shows the basics of how to use the xml library:

int OpenXML(const char\* XMLFile)

{

//Load XML

XML\* xml = new XML(XMLFile);

//Find root element

XMLElement\* Root = xml->GetRootElement();

//Get number of children and a list of all children

unsigned int Cnt = Root->GetChildrenNum();

XMLElement\*\* Child = Root->GetChildren();

//Loop list

for (unsigned int i = 0; i < Cnt; i++)

{

//Search for a specific attribute

XMLVariable\* Att = Child[i]->FindVariableZ("X");

if (Att) //If found just print it to stdout.

{

char Buf[255];

Att->GetValue(Buf);

printf("Attribute X has value %s\n",Buf);

}

}

delete xml;

return 0;

}